UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/755,502	01/05/2001	Arthur H. Khu	X-779 US	5243

24309 7590 12/03/2004

XILINX, INC
ATTN: LEGAL DEPARTMENT
2100 LOGIC DR
SAN JOSE, CA 95124

EXAMINER	
YIGDALL, MICHAEL J	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 12/03/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/755,502	KHU, ARTHUR H.	
	Examiner	Art Unit	
	Michael J. Yigdall	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 August 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's response filed on August 16, 2004 has been fully considered. Claims 1-27 remain pending.

Response to Arguments

2. Applicant's arguments have been fully considered but they are not persuasive.

3. Applicant contends that the cited section of Cooper does not suggest keyword statements or any searching for keyword statements in the program code, and similarly that Cooper does not rely on a keyword statement to determine whether instructions are repeated (Applicant's remarks, page 9, last paragraph).

However, as Applicant acknowledges, Cooper teaches finding sets of repeated instructions by hashing each instruction and entering it into a global table, creating a linear string-like representation of the program, and building a suffix tree. Significantly, Cooper discloses that "each instruction with a unique combination of opcode, registers and constants receives its own table entry, after accounting for semantically identical instructions," and that "each entry in the global table corresponds to an equivalence class of instructions" (page 140, second column, first paragraph).

An instruction, or the opcode of an instruction, is a keyword statement *per se*. Cooper identifies the opcode of each instruction, which is to say that Cooper identifies a keyword statement. Cooper further examines the program code to identify semantically equivalent sets of instructions, based in part on the opcode or keyword statement. In other words, Cooper searches the program code for the opcode or keyword statement to determine whether instructions are

semantically the same. Repeated instructions cannot be identified without considering the opcode or keyword statement.

4. Applicant contends that the limitations of replacing a repeating pattern of statements with an equivalent program loop is not obvious from Cooper because Cooper expressly replaces repeated patterns using either a procedural abstraction or cross-jumping, and that neither the procedural abstraction nor the cross-jumping is suggestive of a program loop (Applicant's remarks, page 10, first paragraph).

However, the examiner disagrees with Applicant's characterization. Applicant states that those skilled in the art will recognize that a program loop includes a control for repeatedly executing the loop until some condition is satisfied, and consequently alleges that neither the procedural abstraction nor the cross-jumping suggests the use of a program loop since there is no control for limiting repetition of the loop (Applicant's remarks, page 10, first paragraph).

Nonetheless, a loop in a program is understood to be "a set of statements in a program executed repeatedly, either a fixed number of times or until some condition is true or false" (Microsoft Press Computer Dictionary, Third Edition, page 290). In the procedural abstraction disclosed by Cooper, repeated patterns of code are replaced with calls to a procedure (page 141, section 3 and Figure 3), so as to channel execution of the repeat through a single copy of the code (page 139, second column, third paragraph). Each procedure is bounded by a call instruction and a return instruction (page 141, second column, third paragraph).

One of ordinary skill in the art would recognize that when a number of repeated patterns occur in succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly. The number of calls and returns

controls the number of iterations. Therefore, after procedural abstraction, the repeated pattern of code operates as a program loop, in which the set of statements in the procedure is executed repeatedly a fixed number of times.

Moreover, although Applicant suggests that a simple jump statement does not by itself construct a program loop (Applicant's remarks, page 10, second paragraph), the examiner maintains that the effect of a jump statement that returns execution back to the start of a set of instructions is analogous to the operation of a program loop.

5. Applicant contends that the limitations of claim 2 include converting each data reference in a keyword statement to a data array reference (Applicant's remarks, page 10, last paragraph), and further that the invention includes changing the program code to include a data array reference in a keyword statement, whereas Cooper does not change the program code to include data array references (Applicant's remarks, page 11, top).

However, the limitation in question recites, "converting the optional data reference, if present, from each located keyword statement to a data array reference" (Applicant's claim 2). The claim does not call for a particular means of converting. Cooper discloses hashing each instruction and entering it into a global table, where each instruction may include registers and constants (page 140, second column, first paragraph). References to registers and constants in an instruction are data references *per se*, and a table is a data array *per se*. Therefore, by hashing each instruction and its data references to enter it into an array, Cooper is converting the data references to data array references.

Furthermore, the feature of "changing the program code to include a data array reference in a keyword statement" is not recited in the claim. As noted above, the claim does not call for a

particular means of converting, much less changing the program code to include a data array reference in a keyword statement. Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

6. Applicant's arguments with respect to claims 3-20 and 21-27 amount to a general allegation that the Office action does not establish a *prima facie* case of obviousness (Applicant's remarks, page 11, second paragraph). Applicant suggests that the limitations of claims 3-20 further refine the limitations of claims 1 and 2. Likewise, as in Applicant's previous response filed on August 16, 2004, which indicated that claims 21-27 were added to claim the invention in alternative language, Applicant again states that claims 21-27 include limitations similar to those of claims 1-20. However, Applicant's arguments with respect to claims 1 and 2 are addressed above, and claims 1-27 are presented accordingly in the claim rejections below.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Enhanced Code Compression for Embedded RISC Processors" by Cooper et al. (art of record, "Cooper" herein).

With respect to claim 1 (original), Cooper discloses a method of optimizing computer program code where the computer program code includes a plurality of statements (see, for example, page 139, the abstract), the method comprising the steps of:

(a) identifying a keyword statement;

(b) searching the program code for the keyword statement; and

(c) determining if the keyword statement begins a repeating pattern of statements in the program code.

Cooper discloses steps (a), (b) and (c) above in terms of finding repeated patterns in the program code by identifying each instruction and its opcode, and searching for equivalent sets of instructions (see, for example, page 140, sections 2 and 2.1). An instruction, or the opcode of an instruction, is a keyword statement *per se*. The equivalent sets of instructions are identified based in part on the opcode, and therefore Cooper identifies keyword statements to find repeated patterns of statements in the program code.

Although Cooper discloses replacing a repeated pattern of statements with a jump instruction or procedure call equivalent to the repeated pattern (see, for example, page 141, section 3, and Figures 3 and 4), Cooper does not expressly disclose the step of:

(d) replacing the repeating pattern of statements with a program loop equivalent to the repeating pattern of statements.

However, when each repeated pattern of statements is replaced with an equivalent procedure call, as taught by Cooper, a corresponding number of call and return instructions are executed (see, for example, page 141, second column, third paragraph). It would have been obvious to one of ordinary skill in the art that when there are a number of repeated patterns in

succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly, in effect operating as a program loop. The number of calls and returns controls the number of iterations.

With respect to claim 2 (original), Cooper further discloses the limitation wherein the keyword statement includes a keyword and an optional data reference, and prior to the searching step:

(a) sequentially locating each keyword statement in the program code; and

(b) converting the optional data reference, if present, from each located keyword statement to a data array reference.

Cooper discloses the limitation and the steps above in terms of locating each instruction or keyword statement, which may include data references such as registers and constants, and converting the instruction and its references to an entry in a table or data array (see, for example, page 140, second column, first paragraph).

With respect to claim 3 (original), Cooper further discloses the limitation wherein the converting includes assigning an array index value to the data array reference where each located keyword statement is assigned a next sequential value of the array index value (see, for example, page 140, second column, second paragraph, which shows that each instruction or keyword statement was assigned an index value into the table or data array).

With respect to claim 4 (original), Cooper further discloses the limitation wherein the determining step further includes:

(a) comparing data array references of two keyword statements from the program code; and

(b) determining if the array index values from the data array references match in size and sequential order.

Coopers discloses the steps above in terms of comparing the entries of two instructions or keyword statements in the table or data array to determine if the two instructions match, i.e. in size and in order, by hashing the instructions and obtaining the index values (see, for example, page 140, second column, first and second paragraphs).

With respect to claim 5 (original), Cooper further discloses the limitation wherein the determining step includes:

(a) determining a first pattern of statements in the program code beginning with a first keyword statement and ending with a statement preceding a second keyword statement that sequentially appears in the program code after the first keyword statement;

(b) determining a second pattern of statements in the program code beginning with the second keyword statement and ending with a statement preceding a third keyword statement that sequentially appears in the program code after the second keyword statement; and

(c) comparing the first pattern of statements to the second pattern of statements; and

(d) setting the first pattern of statements as a repeating pattern if the first and second pattern of statements substantially match.

Cooper discloses the steps above in terms of finding repeated patterns in the program code, which are delineated by first, second, third, etc. instructions or keyword statements, by comparing sets of instructions and determining whether the sets are equivalent (see, for example, page 140, sections 2 and 2.1; also see, for example, page 140, second column, third paragraph

and Figure 2, which show two matching patterns of statements, i.e. a repeated pattern, in the program code).

With respect to claim 6 (original), although Cooper discloses replacing a repeated pattern of statements with a jump instructions or procedure call (see, for example, page 141, section 3, and Figures 3 and 4), Cooper does not expressly disclose the limitation wherein the replacing step includes:

(a) generating loop code for executing a loop within the source code at location of the repeating pattern of statements;

(b) inserting one instance of the repeating pattern of statements within the loop code; and

(c) defining the loop code iterate a number of times equal to a number of instances of the repeating pattern.

However, as set forth above for claim 1, when each repeated pattern of statements is replaced with an equivalent procedure call, as taught by Cooper, a corresponding number of call and return instructions are executed (see, for example, page 141, second column, third paragraph). It would have been obvious to one of ordinary skill in the art that when there are a number of repeated patterns in succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly, in effect operating as a program loop. The number of calls and returns controls the number of iterations.

The call instruction is inserted at the location of the repeated pattern, as in step (a) above (see, for example, page 141, second column, third paragraph), leaving one instance of the repeated pattern in the program code, as in step (b) above (see, for example, page 141, second column, first paragraph). The repeated pattern would iterate the same number of times as the

number of instances of the repeated pattern in the original code (see, for example, page 141, Figure 3), as in step (c) above.

With respect to claim 7 (original), Cooper further discloses the limitation wherein the keyword statement is identified from a predetermined keyword statement (see, for example, page 140, second column, first paragraph, which shows that the instruction or keyword statement is identified based on the opcode, i.e. a predetermined keyword statement from the instruction set).

With respect to claim 8 (original), Cooper further discloses the limitation wherein the keyword statement is identified from a selection made by a user (see, for example, page 140, second column, first paragraph, which shows that the instruction or keyword statement is identified based on the specified registers and constants, i.e. selections made by the user).

With respect to claim 9 (original), Cooper further discloses identifying a plurality of keyword statements and repeating the method for optimizing for each of the plurality keyword statements (see, for example, page 140, sections 2 and 2.1, which shows identifying each instruction or keyword statement and finding all of the repeats in the program).

With respect to claim 10 (original), Cooper discloses a software code optimizer (see, for example, page 139, the abstract) comprising:

(a) analyzing program instructions for analyzing a software code and determining an occurrence of a repeating pattern of code therein (see, for example, page 140, sections 2 and 2.1, which shows analyzing software code to find repeated patterns of instructions).

Although Cooper discloses replacing a repeated pattern of statements with a jump instruction or procedure call to perform an equivalent function as the original code (see, for example, page 141, section 3, and Figures 3 and 4), Cooper does not expressly disclose:

(b) converting program instructions for converting the repeating pattern of code to a programming loop that performs an equivalent function as the repeating pattern of code.

However, when each repeated pattern of statements is replaced with an equivalent procedure call, as taught by Cooper, a corresponding number of call and return instructions are executed (see, for example, page 141, second column, third paragraph). It would have been obvious to one of ordinary skill in the art that when there are a number of repeated patterns in succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly, in effect operating as a program loop. The number of calls and returns controls the number of iterations.

With respect to claim 11 (original), Cooper further discloses the limitation wherein the analyzing program instructions further include:

(a) program instructions for searching the software code for a keyword; and

(b) program instructions for identifying if the keyword begins a repeating pattern of code within the software code and determining a number of occurrences the repeating pattern.

Cooper discloses the features above in terms of identifying repeated patterns of code by searching for instructions or keywords (see, for example, page 140, sections 2 and 2.1), and determining the number of fragments or occurrences of the repeated pattern (see, for example, page 140, second column, third paragraph).

With respect to claim 12 (original), Cooper further discloses program instructions for repeating the analyzing program instructions for a plurality of keywords (see, for example, page 140, sections 2 and 2.1, which shows analyzing each instruction or keyword in the program).

With respect to claim 13 (original), although Cooper discloses replacing a repeated pattern of statements with a jump instruction or procedure call (see, for example, page 141, section 3, and Figures 3 and 4), Cooper does not expressly disclose the limitation wherein the converting program instructions further include program instructions for setting the programming loop to repeat a number of times equal to the number of occurrences of the repeating pattern and inserting one occurrence of the repeating pattern within the programming loop.

However, as set forth above for claim 10, when each repeated pattern of statements is replaced with an equivalent procedure call, as taught by Cooper, a corresponding number of call and return instructions are executed (see, for example, page 141, second column, third paragraph). It would have been obvious to one of ordinary skill in the art that when there are a number of repeated patterns in succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly, in effect operating as a program loop. The number of calls and returns controls the number of iterations.

One occurrence of the repeated pattern is left within the program code (see, for example, page 141, second column, first paragraph), and the repeated pattern would iterate the same number of times as the number of original occurrences (see, for example, page 141, Figure 3).

With respect to claim 14 (original), Cooper further discloses program instructions for locating data references in each statement of program code containing the keyword and

converting the data references to data array references (see, for example, page 140, second column, first paragraph, which shows locating each instruction or keyword statement, which may include data references such as registers and constants, and converting the instruction and its references to an entry in a table or data array).

With respect to claim 15 (original), Cooper further discloses a compiler for translating the software code to an object code executable by a computer (see, for example, page 139, the abstract, which shows an optimizing compiler for translating the software code to executable code).

With respect to claim 16 (original), Cooper discloses a process for optimizing a software code that includes a plurality of statements (see, for example, page 139, the abstract), the process comprising the steps of:

(a) locating multiple occurrences of a code pattern within the software code where the multiple occurrences appear sequentially to each other in the software code (see page 140, sections 2 and 2.1, which show locating repeated patterns of instructions in the software code, i.e. including multiple occurrences appearing sequentially).

Although Cooper discloses replacing a repeated pattern of statements with a jump instruction or procedure call to produce an equivalent result as the original code (see, for example, page 141, section 3, and Figures 3 and 4), Cooper does not expressly disclose the steps of:

(b) generating a program loop that executes one occurrence of the code pattern a number of times to produce an equivalent result as executing the multiple occurrences of the code pattern; and

(c) replacing the multiple occurrences the code pattern the software code with the program loop.

However, when each repeated pattern of statements is replaced with an equivalent procedure call, as taught by Cooper, a corresponding number of call and return instructions are executed (see, for example, page 141, second column, third paragraph). It would have been obvious to one of ordinary skill in the art that when there are a number of repeated patterns in succession, a corresponding number of calls and returns would be executed consecutively, thus iterating through the pattern repeatedly, in effect operating as a program loop. The number of calls and returns controls the number of iterations.

With respect to claim 17 (original), Cooper further discloses:

(a) selecting a keyword statement; and

(b) defining the code pattern based on the keyword statement.

Cooper discloses the steps above in terms of defining repeated patterns in the program code by identifying and analyzing each instruction and its opcode (see, for example, page 140, sections 2 and 2.1). An instruction, or the opcode of an instruction, is a keyword statement *per se*.

With respect to claim 18 (original), Cooper further discloses the limitation wherein the defining step includes:

(a) locating a first instance of the keyword statement in the software code;

(b) defining a first code pattern to include at least the first instance of the keyword statement;

(c) adding subsequent non-keyword statements to the first code pattern until a second instance of the keyword statement appears in the software code;

(d) defining a second code pattern to include at least the second instance of the keyword statement;

(e) adding subsequent non-keyword statements to the second code pattern until a third instance of the keyword statement appears in the software code or until a number of the subsequent non-keyword statements added equal a number of the subsequent non-keyword statements in the first code pattern; and

(f) comparing the first code pattern with the second code pattern to determine if the second code pattern is a multiple occurrence the first code pattern.

Cooper discloses the steps above in terms of finding repeated patterns in the software code, which are delineated by first, second, third, etc. instances of instructions or keyword statements, by comparing sets of instructions and determining whether the sets are equivalent (see, for example, page 140, sections 2 and 2.1, which also shows constructing a suffix tree by adding statements and forming nodes or patterns; also see, for example, page 140, second column, third paragraph and Figure 2, which show two matching patterns of statements, i.e. multiple occurrences of a pattern, in the program code).

With respect to claim 19 (original), Cooper further discloses, prior to the locating step:

(a) selecting at least one keyword statement where the keyword statement includes a keyword and an optional data reference; and

(b) converting each of the data references that appear in each keyword statement in the software code to a data array reference, the data array reference being loaded with values of the converted data references.

Cooper discloses the limitation and the steps above in terms of locating each instruction or keyword statement, which may include data references such as registers and constants, and converting the instruction and its references to an entry in a table or data array (see, for example, page 140, right-hand column, first paragraph).

With respect to claim 20 (original), Cooper further discloses the limitation wherein the generating a program loop step includes generating a looping instruction (see, for example, page 141, section 3, and Figures 3 and 4, which shows replacing a repeated pattern of statements with a jump instruction or procedure call; as set forth above for claim 16, the call and return instructions in effect operate as looping instructions).

With respect to claims 21-27 (previously presented), the limitations of the method and apparatus recited in the claims are similar to the limitations recited in the preceding claims, as indicated by Applicant (see Cooper as applied to claims 1-20 above).

Conclusion

1. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

2. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy

WEI Y. ZHEN
PRIMARY EXAMINER